

**Фісенко Дмитро Михайлович**

аспірант кафедри інфокомунікаційної інженерії імені В.В. Поповського
Харківський національний університет радіоелектроніки, Харків, Україна
ORCID: 0009-0008-4182-3317
dmytro.fisenko@nure.ua

Ільков Андрій Анатолійович

науковий співробітник
Харківський національний університет повітряних сил ім. І. Кожедуба, Харків, Україна
ORCID: 0000-0002-9709-2946
andreyilko428@gmail.com

МОДЕЛЬ КІБЕРЗАГРОЗ КОНТЕЙНЕРІЗОВАНИХ ХМАРНИХ СЕРЕДОВИЩ

Анотація. Контейнеризовані хмарні середовища, керовані оркестраторами класу Kubernetes, мають власну площину керування: API-сервер, службу допуску (admission), планувальник і контролери масштабування. Кожен із цих компонентів утворює окрему поверхню атаки, а відповідні атаки здебільшого не мають виразних ознак на рівні мережеских потоків. У статті розроблено формальну модель кіберзагроз оркестрованого хмарного середовища, яка визначає компоненти інфраструктури, класифікує шість векторів атак інфраструктурного рівня (витік з контейнера, підвищення привілеїв, зловживання API оркестратора, латеральний рух, виснаження ресурсів, атаки на ланцюг постачання образів) і відображає їх на спостережувані телеметричні ознаки. Кожен вектор описано початковими можливостями порушника, передумовами, цільовими компонентами, стадіями та множиною ознак. Сформовано ознаковий простір із 51 ознакою у шести групах, зокрема, з групою ознак автентифікації, що розрізняє підвищення привілеїв і зловживання API. На основі моделі поставлено задачу прогнозування інфраструктурних загроз як виявлення атакувальних епізодів до стадії впливу за заданої частоти хибних спрацювань. Запропоновано метод HMS-AE-AT, що розвиває метод SAE-AT і додає чотири нові складові: багатомасштабну оцінку аномальності для скритних атак, стійкий до отруєння буфера адаптивний поріг, дворівневий детектор з аналізом узгодженості тривоги між просторами імен (він відрізняє планові кластерні зміни від атак і виявляє розподілені атаки) та ознаки автентифікації суб'єктів. Експеримент проведено на навмисно ускладненій, незамкненій телеметрії кластера з 12 просторів імен (5 прогонів; звичайні, скритні та розподілені атаки; дрейф навантаження й нова легітимна подія оновлення кластера). Метод має найвищу серед напівкерованих методів якість ранжування (ROC-AUC 0,881 проти 0,811 у прямого перенесення SAE-AT); дворівневий детектор удвічі знижує повний FPR на оновленні кластера (0,035 проти 0,018) майже без втрати повноти; стійкий поріг не дає повільній атаці підняти поріг (найвищий поріг зростає у 43 рази, стійкий – у 1,12); введення ознак автентифікації зменшило перекриття сигнатур V2 і V3 з 0,43 до 0,29 і підвищило точність атрибуції. Без прикладів вектора в навчанні повнота керованого Random Forest різко падає, тоді як запропонований метод її зберігає. Зазначено компроміс: контекстна нормалізація покращує ранжування й знижує FPR, але послаблює чутливість до найскритніших атак; перевірку виконано на емулятованих, а не на реальних даних.

Ключові слова: кібербезпека; модель загроз; Kubernetes; контейнеризовані хмарні середовища; оркестрування контейнерів; служба допуску; виявлення аномалій; напівкероване навчання.

ВСТУП

Постановка проблеми. Контейнеризація та оркестрування контейнерів (передусім платформою Kubernetes і сумісними з нею керованими сервісами) стали базовою моделлю розгортання хмарних застосунків. Оркестрований кластер є розподіленою системою з власною площиною керування: API-сервером, службою допуску (admission), планувальником, контролерами масштабування та сховищем



стану. Кожен із цих компонентів створює окрему поверхню атаки. Найнебезпечніші атаки на таку інфраструктуру, а саме втеча з контейнера, підвищення привілеїв через рольову модель доступу, зловживання API оркестратора, латеральний рух між подами, виснаження ресурсів і компрометація образів у ланцюгу постачання, часто не мають виразних ознак на рівні окремого мережевого потоку. Натомість вони проявляються в журналах аудиту API, рішеннях служби допуску, подіях масштабування, графі взаємодії компонентів і системних подіях вузла [1], [2].

Методи виявлення мережевих загроз, зокрема напівкерований автоенкодер з адаптивним перцентильним порогом SAE-AT [22], працюють на рівні мережевих потоків і не використовують інфраструктурну семантику кластера. Тому постає науково-практична задача: формально описати, з яких компонентів складається контейнеризоване хмарне середовище, якими векторами його атакують і в яких спостережуваних телеметричних ознаках ці вектори проявляються. Така модель має слугувати основою для побудови ознакового простору методів прогнозування інфраструктурних кіберзагроз і для оцінювання ризиків хмарної інфраструктури.

Аналіз останніх досліджень і публікацій. Моделюванню загроз контейнерних екосистем присвячено роботу [1], у якій методологію STRIDE застосовано до ланцюга «репозиторій коду – збирання образу – реєстр – розгортання» та систематизовано вразливості й контрзаходи. Модель [1] є якісною: вона не встановлює відповідності між загрозами та кількісними ознаками телеметрії, тож її не можна безпосередньо використати для навчання детектора. Подібно, в [2] поєднано аналіз архітектури Kubernetes, відомих CVE та практик посилення захисту площини керування, проте результатом є рекомендації з конфігурування, а не модель виявлення.

Значна частина досліджень зосереджена на статичному аналізі конфігурацій. У [3] на вибірці з 2039 маніфестів Kubernetes із 92 відкритих репозиторіїв виявлено 1051 помилку конфігурації безпеки 11 категорій (відсутність обмежень ресурсів і securityContext, увімкнення hostIPC тощо). У [4] порівняно сканери помилок конфігурації Kubernetes і показано, що їхнє покриття істотно різниться. Статичні засоби потрібні, але вони не фіксують поведінку під час виконання і не виявляють атак, що використовують легітимно допущені об'єкти.

Загрози ланцюга постачання систематизовано в таксономії [5]. Лонгітудне дослідження [6] показало зростання кількості вразливостей в офіційних образах Docker Hub попри їх регулярне оновлення, а широкомасштабне вимірювання [7] охопило безпеку всієї екосистеми образів. Ці роботи описують вразливості образів, але не пов'язують їх із сигналами служби допуску, тобто з тією точкою, де образ потрапляє у кластер.

Для виявлення втечі з контейнера запропоновано провенанс-аналіз [9], граф залежностей системних подій [10] та вибіркву незмінність файлової системи [11]. У [8] продемонстровано міжконтейнерні атаки через механізми eBPF, що підтверджує: межа ізоляції контейнера може бути порушена через спільне ядро вузла. Хостові детектори аномалій на основі системних викликів [12] та eBPF-моніторингу доступу до файлів з автоенкодером [13] досягають високої точності, але працюють лише на рівні вузла й не бачать аномалій у площині керування.

На рівні мережевої взаємодії кластера в [14] імовірнісні скінченні автомати, навчені на NetFlow-даних кластера Kubernetes, виявляють багатокрокові атаки з інтерпретованим результатом. У [15] методи машинного навчання застосовано до метрик кластера для виявлення аномалій продуктивності. У [16] запропоновано багатокласовий детектор мережевих атак у Kubernetes з адаптивним обманом (deception); автори повідомляють точність 87–91% і середню частоту хибних спрацювань 5,5%. Водночас модель [16] навчається в керованому режимі на обмеженому наборі сценаріїв і не порівнюється з базовими детекторами. Федеративна схема [17] поєднує варіаційний автоенкодер та ізоляційний ліс на eBPF-телеметрії кількох кластерів, але механізм вибору порогу в ній не формалізовано. Відкритий набір [18] містить лише метрики ресурсів подів (процесор, пам'ять, диск) і переважно охоплює атаки виснаження ресурсів.

Огляд [19] систематизує методи виявлення аномалій у мікросервісних хмарних застосунках. Більшість цих методів орієнтована на збої продуктивності, а не на атаки. Бенчмарк [20] показує, що жоден метод виявлення аномалій не домінує для всіх типів даних, а якість суттєво залежить від відповідності припущень методу структурі аномалій. Статистичне обґрунтування перцентильного порогу як конформного тесту на викиди наведено в [21]: за умови взаємозамінності калібрувальних і нових нормальних спостережень ймовірність хибного спрацювання обмежена заданим рівнем. Цю властивість використано в методі SAE-AT [22] для мережевого трафіку, а методи моніторингу трафіку та виявлення аномалій у децентралізованих мережах розглянуто в [23].

Аналіз публікацій дає змогу виокремити невирішені частини загальної проблеми:

– відсутня формальна модель, яка в межах одного опису поєднує компоненти оркестрованого кластера, вектори атак інфраструктурного рівня та їхні спостережувані телеметричні прояви; наявні моделі



загроз [1], [2], [5] якісні, а детектори [12]–[17] прив'язані до одного джерела даних;

– не сформовано ознакового простору, що одночасно охоплює аномалії допуску, масштабування та взаємодії між компонентами кластера;

– у напівкерованих детекторах для контейнерних середовищ поріг зазвичай фіксований або неформалізований [16], [17]; не досліджено, чи зберігається аналітичний контроль частоти хибних спрацювань, властивий SAE-AT [22], на неоднорідних і нестационарних інфраструктурних даних;

– бракує відкритих наборів даних з розміщеною інфраструктурною телеметрією площини керування для всіх шести класів векторів атак [14], [18].

Мета статті. Метою статті є розроблення формальної моделі кіберзагроз контейнеризованих (оркестрованих) хмарних середовищ, яка визначає компоненти інфраструктури, класифікує вектори атак інфраструктурного рівня та відображає їх на спостережувані телеметричні ознаки, а також побудова на її основі напівкерованого методу прогнозування інфраструктурних кіберзагроз з адаптивним перцентильним порогом.

Для досягнення мети поставлено такі завдання:

1) розробити формальну модель кіберзагроз оркестрованого хмарного середовища: компоненти інфраструктури (вузли, поди/контейнери, API-сервер оркестратора, служба допуску, мережа взаємодії між компонентами), класифікацію векторів атак (втеча з контейнера, підвищення привілеїв, зловживання API оркестратора, латеральний рух, виснаження ресурсів, атаки на ланцюг постачання образів) та їх відображення на спостережувані інфраструктурні ознаки;

2) поставити задачу прогнозування інфраструктурних кіберзагроз на основі розробленої моделі з урахуванням аномалій допуску, масштабування та взаємодії між компонентами кластера;

3) сформувати ознаковий простір інфраструктурного рівня на основі телеметрії оркестратора, журналів допуску та мережевої взаємодії між компонентами кластера;

4) розробити математичну модель методу, що поєднує напівкероване навчання на нормальній інфраструктурній поведінці та адаптивний перцентильний пороговий механізм і узагальнює принцип методу SAE-AT [22] на інфраструктурні дані;

5) програмно реалізувати метод, сформувати набір даних для експериментальної перевірки, провести експериментальне дослідження, аналіз чутливості до порогу та порівняння з базовими підходами.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

1. Компонентна модель оркестрованого хмарного середовища

Контейнеризоване (оркестроване) хмарне середовище подамо кортежем:

$$\mathcal{K} = \langle N, P, A, M, S, R, U, G \rangle, \quad (1)$$

де $N = \{n_1, \dots, n_{|N|}\}$ – множина вузлів кластера; $P = P(t)$ – множина подів, що існують у момент t ; A – API-сервер оркестратора; M – служба допуску (admission); S – планувальник і контролери, зокрема горизонтальний автомасштабувальник (HPA); R – реєстр (множина джерел) образів контейнерів; U – множина суб'єктів доступу (користувачі, конвеєри CI/CD, сервісні облікові записи); G – динамічний граф мережевої взаємодії між компонентами. Структуру моделі (1) показано на рис. 1.

Кожен вузол $n \in N$ описується вектором ємності ресурсів $\mathbf{c}_n = (c_n^{cpu}, c_n^{mem}, c_n^{pid})$, де c_n^{cpu} – кількість процесорних ядер, c_n^{mem} – обсяг оперативної пам'яті, c_n^{pid} – граничне число процесів. Розміщення подів на вузлах задає відображення $\pi: P \rightarrow N$. Под $p \in P$ подамо кортежем $p = \langle ns(p), img(p), sa(p), \sigma(p), \mathbf{r}(p) \rangle$, де $ns(p) \in C$ – простір імен (контекст) пода, $C = \{c_1, \dots, c_{|C|}\}$ – множина просторів імен; $img(p)$ – образ контейнера; $sa(p) \in U$ – сервісний обліковий запис, токен якого монтується в под; $\mathbf{r}(p)$ – запити та обмеження ресурсів (порожнє значення означає відсутність обмежень); $\sigma(p)$ – контекст безпеки:

$$\sigma(p) = (\sigma^{priv}, \sigma^{hPID}, \sigma^{hNet}, \sigma^{hIPC}, \sigma^{hPath}, \sigma^{root}, Cap(p)), \quad (2)$$

де $\sigma^{priv}, \sigma^{hPID}, \sigma^{hNet}, \sigma^{hIPC}, \sigma^{hPath}, \sigma^{root} \in \{0,1\}$ – ознаки привілейованого режиму, спільного з вузлом простору процесів, мережі, міжпроцесної взаємодії, монтування каталогу вузла та виконання від root; $Cap(p)$ – множина додаткових можливостей ядра Linux (capabilities).

API-сервер A обробляє потік запитів $E^A = \{e\}$, де кожен запис журналу аудиту:

$$e = \langle \tau_e, u_e, vb_e, res_e, ns_e, code_e \rangle, \quad (3)$$

містить час τ_e , суб'єкта $u_e \in U$, дієслово $vb_e \in \{get, list, watch, create, update, patch, delete, exec\}$, тип ресурсу res_e (pods, secrets, roles, rolebindings тощо), простір імен ns_e і код відповіді $code_e$ (наприклад, 200, 403).

Служба допуску M реалізує відображення $\alpha_P: Q \rightarrow \{allow, deny\} \times \Delta$, де Q – множина запитів допуску (AdmissionReview) на створення або зміну об'єктів; P – чинна політика допуску; Δ – множина мутацій (змін специфікації). Запит $q \in Q$ містить специфікацію пода, отже, контекст безпеки s_q та атрибути образу. Образ $i \in R$ характеризується кортежем $i = \langle reg(i), dig(i), sig(i), tag(i), v(i) \rangle$, де $reg(i)$ – реєстр-джерело, $dig(i)$ – криптографічний дайджест, $sig(i) \in \{0,1\}$ – наявність перевіреного підпису, $tag(i)$ – тег (змінний тег на кшталт latest не фіксує вміст), $v(i) \in [0,10]$ – максимальна оцінка CVSS відомих вразливостей образу.

Компонент S визначає бажану кількість реплік $\rho_w(t)$ кожного робочого навантаження w та генерує події планування й масштабування. Мережеву взаємодію у вікні t подамо орієнтованим графом $G_t = (V_t, E_t)$, $V_t = P(t) \cup N \cup \{A, M, etcd, ext\}$, де вершина ext позначає зовнішні щодо кластера адреси, а ребро $(a, b, port, bytes, st) \in E_t$ відповідає потоку від a до b на порт $port$ з обсягом $bytes$ і статусом встановлення з'єднання st . Базовий граф нормальної взаємодії контексту c позначено $E_c^{base} = \bigcup_{t \in T_{train}} E_{c,t}$.

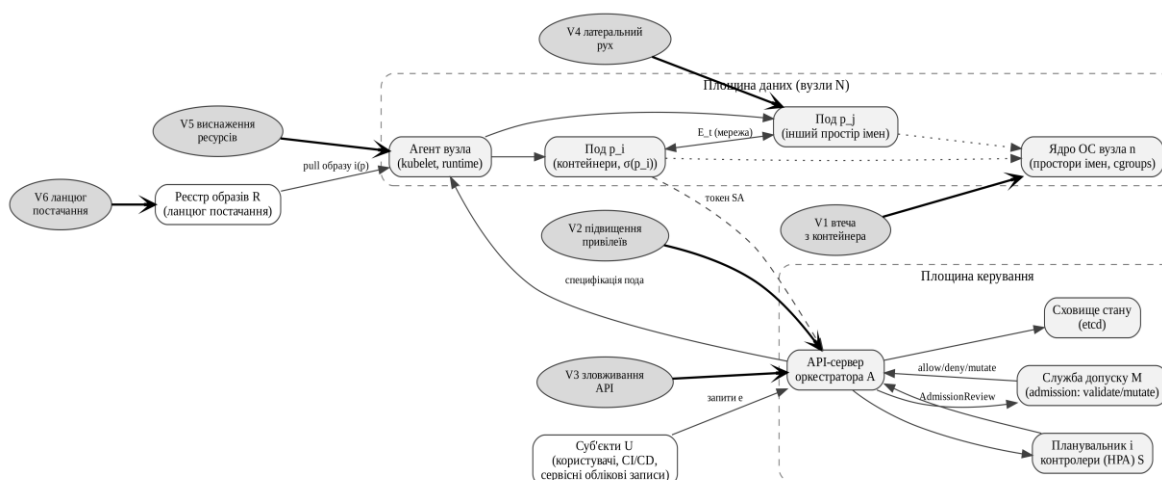


Рис. 1. Компонентна модель оркестрованого хмарного середовища та точки прикладання векторів атак V1–V6

2. Класифікація векторів атак інфраструктурного рівня

Порушник моделюється як суб'єкт з однією з початкових можливостей $\theta = \{\theta_{pod}, \theta_{cred}, \theta_{img}\}$: виконання коду всередині скомпрометованого контейнера (θ_{pod}), володіння викраденими обліковими даними чи токеном (θ_{cred}) або можливість опублікувати шкідливий образ у джерелі, з якого кластер завантажує образи (θ_{img}). Вектор атаки v_k , $k = 1, \dots, K$, $K = 6$, визначається кортежем $v_k = \langle \theta_k, Pre_k, Tgt_k, P_k, Eff_k, \Phi_k \rangle$, де $\theta_k \subseteq \theta$ – необхідні початкові можливості; $Pre_k(\mathcal{K}) \in \{0,1\}$ – предикат передумов на конфігурації кластера; $Tgt_k \subseteq \{N, P, A, M, S, R, G\}$ – цільові компоненти; $P_k = (prep, expl, imp)$ – послідовність стадій (підготовка/розвідка, експлуатація, вплив); $Eff_k \subseteq \{C, I, Av\}$ – порушувані властивості (конфіденційність, цілісність, доступність); $\Phi_k \subseteq F$ – множина спостережуваних ознак, у яких проявляється вектор. Класифікацію наведено в табл. 1.

Передумови Pre_k можна обчислити статично з конфігурації кластера (маніфестів, ролей, мережних політик), тоді як прояви Φ_k спостерігаються лише під час виконання. Це розділення робить модель придатною для двох задач: оцінювання ризику за конфігурацією та виявлення атак за телеметрією. Для оцінювання ризику кластера в моделі є вираз:

$$Risk(\mathcal{K}) = \sum_{k=1}^K \omega_k \cdot Pre_k(\mathcal{K}) \cdot (1 - \hat{r}_k) \cdot I_k, \quad (4)$$

де $\omega_k \in [0,1]$ – апіорна ймовірність застосування вектора v_k (наприклад, з даних розвідки загроз); I_k – експертна оцінка шкоди; $\hat{r}_k$ – оцінена повнота виявлення вектора v_k наявним детектором. Отже, вираз (4) пов'язує конфігураційну експозицію з ефективністю виявлення.



Таблиця 1

Класифікація векторів атак інфраструктурного рівня				
Вектор	θ_k	Передумова $Pre_k(\mathcal{K})$	Tgt_k	Eff_k
V1 Втеча з контейнера	θ_{pod}	$\exists p: \sigma^{priv} \vee \sigma^{hPID} \vee \sigma^{hPath} \vee CAP_SYS_ADMIN \in Cap(p)$ або вразливе середовище виконання	$P \rightarrow N$	C, I, Av
V2 Підвищення привілеїв	$\theta_{pod}, \theta_{cred}$	права $sa(p)$ на create/bind/escalate для roles, rolebindings або get для secrets	A, U	C, I
V3 Зловживання API оркестратора	θ_{cred}	доступність A з анонімним доступом або з токеном, що витік	A, S	C, I, Av
V4 Латеральний рух	θ_{pod}	відсутність мережових політик між просторами імен (дозвіл за замовчуванням)	G, P	C, I
V5 Виснаження ресурсів	$\theta_{pod}, \theta_{img}$	$r(p) = \emptyset$ або відсутність квот простору імен	N, S	Av
V6 Ланцюг постачання образів	θ_{img}	Π не перевіряє $sig(i)$, допускає змінні теги чи недовірені $reg(i)$	R, M, P	C, I, Av

3. Відображення векторів атак на спостережувані ознаки

Телеметрію агрегуємо за контекстами $c \in C$ (просторами імен) і неперекривними часовими вікнами $W_t = [t\Delta, (t+1)\Delta)$ тривалістю Δ . Оператор спостереження:

$$\mathbf{x}_{c,t} = \Omega(E_{c,t}^A, Q_{c,t}, S_{c,t}, G_t, RT_{c,t}) \in \mathbb{R}^D, \quad (5)$$

перетворює записи аудиту API $E_{c,t}^A$, запити допуску $Q_{c,t}$, події масштабування $S_{c,t}$, граф взаємодії G_t і події середовища виконання вузла $RT_{c,t}$ контексту c у вікні W_t на вектор ознак розмірності D . Простір ознак розбито на шість груп $F = F_{API} \cup F_{ADM} \cup F_{SCL} \cup F_{NET} \cup F_{RT} \cup F_{IDN}$, $|F| = D = 51$, де F_{API} – ознаки API-сервера (9 ознак), F_{ADM} – ознаки служби допуску (12), F_{SCL} – ознаки масштабування, планування та ресурсів (7), F_{NET} – ознаки мережової взаємодії (6), F_{RT} – ознаки середовища виконання вузла (3), F_{IDN} – ознаки автентифікації суб'єктів (2). Групу F_{IDN} уведено, щоб розрізняти вектори, які інакше майже збігаються за ознаками: підвищення привілеїв (V2) супроводжується самопідвищенням прав, а зловживання API (V3) – зверненнями від анонімних токенів і службових подів. Аномалії допуску, масштабування та взаємодії між компонентами, визначені в задачі дослідження, відповідають групам F_{ADM}, F_{SCL} і F_{NET} ; групи F_{API}, F_{RT} і F_{IDN} є допоміжними.

Відображення векторів атак на ознаки задамо бінарною матрицею $\Phi \in \{0,1\}^{K \times D}$, де $\Phi_{kj} = 1$, якщо ознака $f_j \in \Phi_k$, і матрицею сили прояву $\kappa^{(s)} \in \mathbb{R}_{\geq 0}^{K \times D}$ для кожної стадії $s \in \{prep, expl, imp\}$. Модель прояву вектора v_k на стадії s у контексті c має вигляд:

$$\mathbb{E}[\mathbf{x}_{c,t} | v_k, s] = \boldsymbol{\mu}_c(t) + \lambda(\kappa_k^{(s)} \odot \mathbf{m} \odot \boldsymbol{\zeta}_c), \quad m_j \sim \text{Bernoulli}(q), \quad (6)$$

де $\boldsymbol{\mu}_c(t)$ – математичне сподівання нормальної поведінки контексту c у вікні t (з урахуванням добової сезонності); $\lambda > 0$ – інтенсивність атаки (менші значення відповідають скритнішій атаці); $\kappa_k^{(s)}$ – k -й рядок матриці сили прояву; $\boldsymbol{\zeta}_c$ – вектор масштабів природної мінливості ознак контексту c ; $\mathbf{m} \in \{0,1\}^D$ – маска часткової спостережуваності: кожна ознака із сигнатури проявляється з імовірністю q ; $\odot$ – поелементний добуток. Модель (6) враховує, що конкретний екземпляр атаки зазвичай проявляється лише в частині ознак свого вектора, а не в усіх.

Перелік ознак наведено в табл. 2, а матрицю відображення Φ з розподілом за стадіями показано на рис. 2. Кожна ознака обчислюється за записами вікна; наведемо формули для характерних ознак. Частка заборонених запитів до API:

$$f_3 = \frac{|e \in E_{c,t}^A: code_e = 403|}{\max(1, |E_{c,t}^A|)}. \quad (7)$$

Ентропія типів ресурсів, до яких звертаються суб'єкти:

$$f_7 = -\sum_r \hat{p}_r \ln \hat{p}_r, \quad \hat{p}_r = \frac{|e \in E_{c,t}^A: res_e = r|}{|E_{c,t}^A|}. \quad (8)$$

Кількість запитів допуску на створення привілейованих подів $f_{13} = |\{q \in Q_{c,t}: \sigma_q^{priv} = 1\}|$. Частка нових ребер графа взаємодії та віялове розходження (кількість різних адресатів) $f_{37} = \frac{|E_{c,t} \setminus E_{c,t}^{base}|}{\max(1, |E_{c,t}|)}$, $f_{36} = |\{b: (a, b) \in E_t, a \in P_c\}|$, де $E_{c,t} \subseteq E_t$ – ребра, що виходять з подів P_c контексту c . Ентропія портів призначення f_{42} обчислюється аналогічно (8) за розподілом портів. Частки $(f_2, f_3, f_8, f_9, f_{11}, f_{12}, f_{17-19}, f_{21}, f_{29}, f_{37}, f_{38}, f_{43})$ нормовано на кількість подій вікна, а коефіцієнти завантаження f_{26-28} – на ємність



ресурсу. Лічильники ($f_1, f_4-f_6, f_{10}, f_{13}-f_{16}, f_{22}-f_{25}, f_{30}-f_{32}, f_{34}, f_{36}, f_{39}, f_{40}, f_{44}-f_{48}$) подано абсолютними значеннями за вікно Δ . Неперервні ознаки $f_7, f_{20}, f_{33}, f_{35}, f_{41}, f_{42}$ є додатними дійсними величинами.

Таблиця 2

Ознаковий простір інфраструктурного рівня

Група	Ознаки (позначення: зміст)
F_{API} , аудит API-сервера	f_1 – інтенсивність запитів; f_2 – частка модифікувальних дієслів; f_3 – частка відповідей 403; f_4 – читання secrets; f_5 – зміни RBAC (roles, bindings); f_6 – exec/attach/port-forward; f_7 – ентропія типів ресурсів; f_8 – частка нових суб'єктів; f_9 – частка list/watch
F_{ADM} , служба допуску	f_{10} – інтенсивність запитів допуску; f_{11} – частка відмов; f_{12} – частка мутацій; f_{13} – привілейовані поди; f_{14} – hostPID/hostNetwork/hostIPC; f_{15} – hostPath; f_{16} – додані capabilities; f_{17} – частка невідписаних образів; f_{18} – частка недовірених реєстрів; f_{19} – частка змінних тегів; f_{20} – середня оцінка CVSS образів; f_{21} – частка раніше небачених образів
F_{SCL} , масштабування і ресурси	f_{22}, f_{23} – створення/видалення подів; f_{24} – зміна кількості реплік; f_{25} – події НРА; f_{26}, f_{27} – середнє та 95-й перцентиль завантаження CPU; f_{28} – завантаження пам'яті; f_{29} – частка тротлінгу CPU; f_{30} – OOM-завершення; f_{31} – перезапуски; f_{32} – поди в стані Pending; f_{33} – кількість процесів
F_{NET} , мережева взаємодія	f_{34} – кількість потоків; f_{35} – вихідний обсяг; f_{36} – віялове розходження; f_{37} – частка нових ребер; f_{38} – частка міжпросторових потоків; f_{39} – звернення подів до API-сервера; f_{40} – звернення до kubelet/etcd; f_{41} – зовнішній вихідний трафік; f_{42} – ентропія портів; f_{43} – частка невдалих з'єднань; f_{44} – інтенсивність DNS-запитів
F_{RT} , середовище виконання	f_{45} – чутливі системні виклики (mount, setns, unshare, ptrace); f_{46} – записи у файлову систему вузла; f_{47} – процеси поза простором імен контейнера; f_{48} – запуски командної оболонки
F_{IDN} , автентифікація	f_{49} – частка змін RBAC, що підвищують власні права (самопідвищення); f_{50} – частка запитів від анонімних чи нерозпізнаних токенів; f_{51} – частка звернень до API від службових подів

Матриця Φ має такі властивості. Вектори покривають 48 із 51 ознаки; ознаки f_{11}, f_{12}, f_{34} не входять до жодної сигнатури і виконують роль контекстних, тобто описують нормальний режим. Потужність сигнатур $|\Phi_k|$ становить 12, 10, 12, 8, 14 і 13 ознак для V1–V6 відповідно. Найбільше перекриття (коефіцієнт Жаккара $J = 0,29$) мають вектори V2 і V3: обидва проявляються в аудиті API-сервера. Завдяки введенню ознак автентифікації F_{IDN} це перекриття зменшено (без них воно становило б $J = 0,43$). Перекриття V4 і V6 дорівнює $J = 0,24$ (спільні мережеві прояви), решта пар мають $J \leq 0,17$. Кількість унікальних ознак, що належать лише одному вектору, становить 7, 3, 5, 4, 9 і 4 відповідно. Отже, після введення групи F_{IDN} кожен вектор має власні відмітні ознаки, а найбільша залишкова неоднозначність – між V2 і V3.

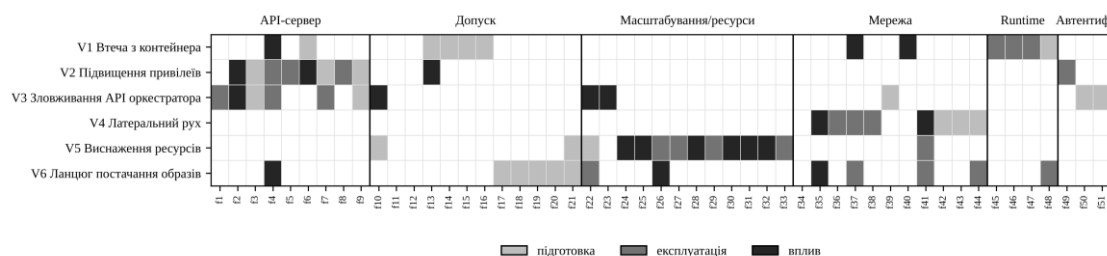


Рис. 2. Матриця відображення Φ векторів атак V1–V6 на ознаки f_1 – f_{51} (відтінок – стадія першої появи ознаки)

Матриця Φ має такі властивості. Вектори покривають 48 із 51 ознаки; ознаки f_{11}, f_{12}, f_{34} не входять до жодної сигнатури і виконують роль контекстних, тобто описують нормальний режим. Потужність сигнатур $|\Phi_k|$ становить 12, 10, 12, 8, 14 і 13 ознак для V1–V6 відповідно. Найбільше перекриття (коефіцієнт Жаккара $J = 0,29$) мають вектори V2 і V3: обидва проявляються в аудиті API-сервера. Завдяки



введенню ознак автентифікації F_{IDN} це перекриття зменшено (без них воно становило б $J = 0,43$). Перекриття V4 і V6 дорівнює $J = 0,24$ (спільні мережеві прояви), решта пар мають $J \leq 0,17$. Кількість унікальних ознак, що належать лише одному вектору, становить 7, 3, 5, 4, 9 і 4 відповідно. Отже, після введення групи F_{IDN} кожен вектор має власні відмітні ознаки, а найбільша залишкова неоднозначність – між V2 і V3.

4. Постановка задачі прогнозування інфраструктурних кіберзагроз

Кожному спостереженню $\mathbf{x}_{c,t}$ відповідає прихована під час навчання мітка $y_{c,t} \in \{0,1\}$: $y_{c,t} = 1$, якщо у вікні W_t контексту c діє будь-яка стадія якогось вектора атаки, і $y_{c,t} = 0$ інакше. Прихований індекс вектора позначимо $k_{c,t}$. Атакуючий епізод ε – це послідовність вікон $[t_\varepsilon^s, t_\varepsilon^e]$ зі стадіями підготовки, експлуатації та впливу; початок стадії впливу позначимо t_ε^{imp} . Епізод може охоплювати один або кілька просторів імен. Модель навчають лише на нормальній поведінці $\mathcal{D}_N = \{\mathbf{x}_{c,t} : y_{c,t} = 0, t \in T_{train}\}$, $\mathcal{D}_{cal} = \{\mathbf{x}_{c,t} : y_{c,t} = 0, t \in T_{cal}\}$, де $T_{train} < T_{cal} < T_{test}$ – послідовні в часі інтервали навчання, калібрування й тесту (дані з майбутнього не використовують). Потрібно побудувати оцінку аномальності s і вирішальне правило $\hat{y}_{c,t} = \mathbb{I}[\hat{s}_{c,t} > \tau_c(t)]$ з порогом $\tau_c(t)$, що розв'язують задачу мінімізації $\min \Pr(\hat{y}_{c,t} = 0 \mid y_{c,t} = 1)$ за умови $\Pr(\hat{y}_{c,t} = 1 \mid y_{c,t} = 0) \leq \alpha = 1 - P/100$, де α – допустима частота хибних спрацювань (FPR), яку оператор задає через робочий перцентиль P . Задача є прогнозувальною: епізод треба виявити до початку стадії впливу. Для епізоду з першою тривоگوю в момент t_ε^a вводимо час до виявлення $TTD_\varepsilon = t_\varepsilon^a - t_\varepsilon^s$, випередження $Lead_\varepsilon = t_\varepsilon^{imp} - t_\varepsilon^a$ та ознаку раннього виявлення $\mathbb{I}[t_\varepsilon^a < t_\varepsilon^{imp}]$. Частка епізодів, виявлених до стадії впливу, є головною прогнозувальною метрикою.

5. Метод HMS-AE-AT

Запропонований метод названо HMS-AE-AT (Hierarchical Multi-Scale AutoEncoder with Adaptive Threshold – ієрархічний багатомасштабний автоенкодер з адаптивним порогом). Він розвиває метод SAE-AT [22], у якому автоенкодер навчають лише на нормальних даних, а поріг калібрують за перцентилем окремої нормальної вибірки. Для інфраструктурних даних, які неоднорідні, нестационарні й розподілені між багатьма просторами імен, цього принципу недостатньо. Тому метод побудовано із шести кроків: кроки 1–3 переносять ідею SAE-AT на інфраструктурну телеметрію, а кроки 4–6 – нові складові (багатоканально-багатомасштабна оцінка, стійкий поріг і дворівневий детектор). Кроки 1–5 працюють у межах одного простору імен, крок 6 – на рівні всього кластера. Нижче описано кожен крок простими словами й формулами.

Крок 1. Нормалізація з урахуванням контексту. Ознаки мають різні масштаби й важкі хвости, а «норма» залежить від простору імен (у системних просторах привілейовані поди – це норма). Лічильники та великі величини спершу стискаємо: $\tilde{x}_j = \ln(1 + x_j)$. Потім кожену ознаку в кожному просторі імен c стандартизуємо робастно:

$$z_{c,t,j} = \text{clip}\left(\frac{\tilde{x}_{c,t,j} - \text{med}_{c,j}}{s_{c,j}} - z_{\max}, z_{\max}\right), \quad s_{c,j} = \max\left(\frac{IQR_{c,j}}{1,349}, sd_{c,j}, \delta_j\right), \quad (9)$$

де $\text{med}_{c,j}$, $IQR_{c,j}$, $sd_{c,j}$ – медіана, міжквартильний розмах і стандартне відхилення ознаки j у контексті c , обчислені лише на $\mathcal{D}_N$ (без витоку інформації з калібрувальних і тестових даних); 1,349 – коефіцієнт, що узгоджує IQR зі стандартним відхиленням нормального розподілу; δ_j – нижня межа масштабу, що запобігає діленню на нуль для ознак, сталих у контексті: $\delta_j = 0,35$ для лічильників (одна подія дає $z \approx 2$), $\delta_j = 0,01$ для часток, $\delta_j = 0,05 \cdot |\text{med}_j|$ для неперервних ознак; $z_{\max} = 20$ – межа відсікання.

Крок 2. Автоенкодер нормальної поведінки. Автоенкодер $f_\theta = g_\phi \circ h_\psi$ складається з енкодера h_ψ і декодера g_ϕ та стискає вхід $\mathbb{R}^{51}$ у латентний простір $\mathbb{R}^{12}$ і відновлює назад за архітектурою $51 \rightarrow 64 \rightarrow 32 \rightarrow 12 \rightarrow 32 \rightarrow 64 \rightarrow 51$:

$$\mathbf{h}^{(l)} = \text{ReLU}(\mathbf{W}^{(l)} \mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \quad l = 1, \dots, 5; \quad \hat{\mathbf{z}} = \mathbf{W}^{(6)} \mathbf{h}^{(5)} + \mathbf{b}^{(6)}, \quad (9)$$

де $\mathbf{h}^{(0)} = \mathbf{z}$ – нормалізований вхідний вектор; $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$; $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ – вагова матриця та вектор зсуву шару l розмірності d_l ; $\text{ReLU}(a) = \max(0, a)$; $\mathbf{h}^{(3)} \in \mathbb{R}^{12}$ – латентне подання («вузьке місце»). На відміну від SAE-AT [22], де вихідний шар сигмоїдний, а вхід нормалізовано у $[0,1]$, тут вихідний шар лінійний, оскільки робастно стандартизовані ознаки (9) не обмежені одиничним інтервалом. Загальна кількість параметрів $|\theta| = \sum_{l=1}^6 d_l (d_{l-1} + 1) = 11\,260$. Параметри знаходяться мінімізацією регуляризованої середньоквадратичної помилки на $\mathcal{D}_N$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \frac{1}{|\mathcal{D}_N|D} \sum_{\mathbf{z} \in \mathcal{D}_N} \|\mathbf{z} - f_\theta(\mathbf{z})\|_2^2 + \gamma \sum_l \|\mathbf{W}^{(l)}\|_F^2, \quad (10)$$

де γ – коефіцієнт L_2 -регуляризації, $\|\cdot\|_F$ – норма Фробеніуса. Задачу (10) розв'язують оптимізатором Adam з ранньою зупинкою за відкладеною частиною $\mathcal{D}_N$. Після навчання нормальні вікна відновлюються



з малою помилкою, а незвичні – з великою.

Крок 3. Розріджена оцінка аномальності. Атаки, згідно з (6), проявляються в невеликій підмножині ознак, тому середня помилка по всіх ознаках (як у SAE-AT) розмиває сигнал. Поознакові помилки нормуються на середню помилку кожної ознаки на калібрувальній вибірці, щоб зробити ознаки порівнянними $e_j(\mathbf{z}) = (z_j - \hat{z}_j)^2$, $\tilde{e}_j(\mathbf{z}) = \frac{e_j(\mathbf{z})}{\bar{e}_j}$, $\bar{e}_j = \frac{1}{|\mathcal{D}_{cal}|} \sum_{\mathbf{z} \in \mathcal{D}_{cal}} e_j(\mathbf{z})$. Оцінкою одного вікна (функцією аномальності) беремо середнє m найбільших нормованих помилок:

$$s(\mathbf{z}) = \frac{1}{m} \sum_{i=1}^m \tilde{e}_{(D-i+1)}(\mathbf{z}), \quad (11)$$

де $\tilde{e}_{(1)} \leq \dots \leq \tilde{e}_{(D)}$ – варіаційний ряд нормованих помилок, m – кількість ознак, що враховуються ($1 \leq m \leq D$). Часову узгодженість багатокрокової атаки враховує експоненційне згладжування в межах кожного контексту:

$$\tilde{s}_{c,t} = \beta s(\mathbf{z}_{c,t}) + (1 - \beta) \tilde{s}_{c,t-1}, \quad \beta \in (0,1], \quad (12)$$

де β – коефіцієнт згладжування ($\beta = 1$ означає відсутність згладжування).

Крок 4. Багатоканально-багатомасштабна оцінка. Ця складова розв'язує дві проблеми. Перша – часова: коротка скритна атака дає кілька слабких піків, а повільна – тривалий невеликий підйом, тож одне згладжування не ловить обидва випадки. Тому оцінку згладжують з кількома коефіцієнтами $\beta \in \mathcal{B} = \{1, 0; 0,5; 0,2\}$ ($\beta = 1$ – без згладжування). Друга проблема – нормувальна: контекстна нормалізація (крок 1) добре виявляє відхилення від норми простору імен, але скритна атака навмисно тримається в межах цієї норми й стає непомітною. Тому поряд із контекстним каналом використовують *глобальний* канал – той самий автоенкодер і оцінку, але з нормалізацією (9) по всьому кластеру, а не за кожним простором імен окремо. Позначимо оцінку одного вікна в каналі $ch \in \{ctx, glob\}$ як $s^{ch}(\mathbf{z})$. Підсумкова оцінка – максимум по всіх каналах і шкалах, кожен з яких нормовано на її калібрувальну медіану:

$$s_{c,t}^{ch,\beta} = \beta s^{ch}(\mathbf{z}_{c,t}) + (1 - \beta) s_{c,t-1}^{ch,\beta}, \quad \tilde{s}_{c,t} = \max_{ch, \beta \in \mathcal{B}} \frac{s_{c,t}^{ch,\beta}}{\text{med}_{\{s_{c,t'}^{ch,\beta} : t' \in T_{cal}\}}}. \quad (13)$$

Короткі піки «ловить» шкала $\beta = 1$, повільний підйом – шкала $\beta = 0,2$, а атаки, що тримаються в межах норми контексту, – глобальний канал; завдяки нормуванню всі складові порівнянні між собою.

Крок 5. Адаптивний поріг, стійкий до отруєння. Для кожного контексту c на калібрувальних оцінках $\{\tilde{s}_{c,t}\}_{t \in T_{cal}}$ обсягом n_c початковий поріг визначаємо як скінченновибірковий (конформний) P -й перцентиль:

Для кожного простору імен початковий поріг – це конформний (скінченновибірковий) P -й перцентиль калібрувальних оцінок:

$$\tau_c^{(0)}(P) = \tilde{s}_{c,(k_c)}, \quad k_c = \lceil (n_c + 1)P/100 \rceil, \quad (14)$$

Якщо калібрувальні й нові нормальні оцінки взаємозамінні, то для нормального вікна $\Pr(\tilde{s}_{c,t} > \tau_c^{(0)} | y_{c,t} = 0) \leq 1 - P/100$ [21], тобто оператор прямо керує FPR через P . Навантаження з часом зростає, тому поріг оновлюють кожні u вікон за ковзним буфером $B_c(t)$ останніх L оцінок. Щоб повільна наростаюча атака не «підняла» поріг і не осліпила детектор (отруєння буфера), у буфер додають лише вікна, оцінка яких не перевищує $\eta \tau_c(t)$ (тобто не схожі на тривогу), а дрейф порогу обмежують:

$$B_c(t) = \{\tilde{s}_{c,t'} : t' \leq t, \tilde{s}_{c,t'} \leq \eta \tau_c(t')\},$$

$$\tau_c(t) = \text{clip} \left(\hat{Q}_P(B_c(t)), \tau_c^{(0)} / (1 + \rho), \tau_c^{(0)} (1 + \rho) \right), \quad (15)$$

де $\hat{Q}_P$ – конформний перцентиль (14) буфера; u – період перерахунку у вікнах; $\eta = 1,5$ – поріг допуску у буфер; $\rho \geq 0$ – межа відносного дрейфу порогу, що обмежує можливість «отруєння» порогу повільною атакою (при $\rho = 0$ поріг статичний). Рішення для кожного вікна:

$$\hat{y}_{c,t}^{ns} = \mathbb{I}[\tilde{s}_{c,t} > \tau_c(t)]. \quad (16)$$

Крок 6. Дворівневий детектор: узгодженість між просторами імен. Оцінки (16) бачать лише один простір імен, тому планове оновлення кластера, що одночасно підіймає всі простори імен, спричиняє сплеск хибних тривог, а атака, розподілена між кількома просторами імен, в кожному окремо виглядає надто слабкою. Другий рівень дивиться на весь кластер у кожному вікні. Нехай частка просторів імен, що спрацювали у вікні t , дорівнює:

$$r(t) = \frac{1}{|C|} \sum_{c \in C} \hat{y}_{c,t}^{ns}. \quad (17)$$

Якщо $r(t) \geq \theta_{sys}$, подія є кластерною легітимною зміною (оновлення, обслуговування вузлів), тому всі тривоги цього вікна знімають:

$$\hat{y}_{c,t} = 0 \quad \text{для всіх } c, \text{ якщо } r(t) \geq \theta_{sys}. \quad (18)$$

Навпаки, розподілену атаку шукають серед просторів імен із підвищеною, але підпороговою



оцінкою. Позначимо множину таких «тихих» просторів імен у вікні:

$$A(t) = \{c: \gamma_e \tau_c(t) < \tilde{s}_{c,t} \leq \tau_c(t)\}. \quad (19)$$

Якщо $|A(t)| \geq k_{\min}$ протягом w підряд вікон і це вікно не придушене правилом (18), простори імен з $A(t)$ позначають як узгоджену розподілену аномалію:

$$\hat{y}_{c,t} = 1 \quad \text{для } c \in A(t), \text{ якщо } |A(t')| \geq k_{\min} \forall t' \in [t - w + 1, t] \text{ і } r(t) < \theta_{\text{sys}}. \quad (20)$$

Остаточне рішення $\hat{y}_{c,t}$ – результат застосування (18) і (20) до (16).

Атрибуція вектора. Для виявленого епізоду обчислюємо середній вектор нормованих помилок $\bar{e}_\varepsilon$ за вікнами з тривоги і обираємо оцінку належності до кожного вектора за матрицею відображення Φ :

$$\hat{k}(\varepsilon) = \underset{k}{\operatorname{argmax}} \frac{\sum_j \Phi_{kj} \bar{e}_{\varepsilon,j}}{\sum_j \Phi_{kj}}, \quad (21)$$

Детектор навчається без жодної інформації про атаки, а модель використовується лише для інтерпретації сигналу тривоги. Метод SAE-AT [22] є окремим випадком HMS-AE-AT: один контекст ($|C| = 1$), функція аномальності – середнє ненормованих помилок за всіма ознаками ($m = D$, $\bar{e}_j \equiv 1$), без згладжування ($\beta = 1$) і зі статичним порогом ($\rho = 0$).

МЕТОДИКА ДОСЛІДЖЕННЯ

1. Формування набору даних

Відкриті набори даних для контейнерних середовищ містять або лише мережеві потоки [14], або лише ресурсні метрики подів [18]. Жоден із них не охоплює одночасно журналів аудиту API, рішень служби допуску, подій масштабування та графа взаємодії для всіх шести векторів V1–V6. Тому для перевірки сформовано набір даних за допомогою стохастичного емулятора інфраструктурної телеметрії кластера. Емулятор реалізує оператор спостереження (5). Такий набір дає змогу порівняти методи за контрольованих умов і перевірити, чи узгоджується поведінка методу з моделлю загроз, але не замінює валідації на реальному кластері (див. підрозділ «Обговорення результатів»).

Щоб оцінювання не було замкненим, процес породження атак навмисно зроблено структурно відмінним від матриці відображення Φ , яку метод використовує лише для атрибуції. По-перше, інтенсивність атаки керується латентним фактором з інерцією (процес AR(1)), а не незалежними значеннями у кожному вікні. По-друге, окрім ознак сигнатури, кожна атака зачіпає 1–3 випадкові *побічні* ознаки поза сигнатурою та злегка підвищує навантаження (f_1, f_{34}, f_{26}). По-третє, кожна ознака сигнатури проявляється лише з імовірністю $q = 0,7$. Отже, детектор і атрибуція працюють із даними, що містять шум і побічні сигнали, яких у Φ немає; детектор при цьому взагалі не бачить міток.

Емульований кластер містить $|C| = 12$ просторів імен п'яти типів: системні (kube-system, monitoring, ingress), CI/CD (ci-cd), розробницькі (dev-a, dev-b), продуктивні (prod-web, prod-api, prod-db, payments) і пакетні (analytics, batch-jobs). Профіль кожного простору імен задано базовими інтенсивностями ознак, випадково вибраними з діапазонів, специфічних для типу. Наприклад, у системних просторах імен легітимно присутні привілейовані поди, звернення до kubelet/etcd і чутливі системні виклики; у розробницьких підвищено частоту exec/attach, запусків оболонки та частку змінних тегів. Нормальне навантаження має добову сезонність:

$$L_c(t) = b_c \left(1 + a_c \sin(2\pi(t/1440 - \varphi_c))\right) e^{\xi_t} (1 + \eta t/1440), \quad \xi_t \sim \mathcal{N}(0; 0, 1^2), \quad (22)$$

де $b_c \in [0,6; 1,4]$ – базовий рівень; a_c – амплітуда добового коливання ($a_c \in [0,2; 0,5]$, для пакетних просторів $[0,6; 0,9]$); $\varphi_c \in [0,1]$ – фаза; ξ_t – мультиплікативний шум; η – добовий дрейф. Лічильники генеруються за розподілом Пуассона з інтенсивністю, пропорційною $L_c(t)$, частки – за бета-розподілом, неперервні величини – з логнормальним або нормальним шумом.

В усі розбиття додано легітимні «важкі» події, схожі на аномалії: розгортання (0,5–6 на добу, 3–8 вікон), сплески навантаження з реакцією НРА (0,3–2 на добу, 10–40 вікон), сеанси налагодження (1 на добу, 5–20 вікон) та обслуговування вузлів (drain, 10–30 вікон). У тестовому розбитті додатково є подія, відсутня в навчальних даних, – оновлення кластера тривалістю 90 вікон, що одночасно зачіпає всі простори імен, і дрейф навантаження $\eta = 0,05$ на добу.

Атакувальні епізоди трьох видів вбудовані в тестове розбиття. *Звичайні* (single) – в один простір імен; для кожного з 6 векторів і 3 рівнів інтенсивності по 5 епізодів. *Скритні* (mimicry) – атака тримається нижче типового рівня (інтенсивність удвічі менша) і обходить ознаки рівня вузла; для кожного вектора по 3 епізоди низької і 3 середньої інтенсивності. *Розподілені* (dist) – одна атака одночасно в 3–4 просторах імен, тому в кожному з них вона слабка; для векторів V3 і V4 по 8 епізодів. Інтенсивність λ вимірюється в масштабах природної мінливості ознаки й вибирається з діапазонів [1; 2], [2; 4], [4; 8] для низької, середньої та високої інтенсивності. Тривалості стадій: підготовка 4–8, експлуатація 6–15, вплив 6–15 вікон.



Розбиття впорядковані в часі: навчальне – 6 діб (103 680 вікон), калібрувальне – 1 доба (17 280), тестове – 3 доби (51 840, у середньому 9,6 % атакуючих вікон, 142 епізодів). Для керованого базового методу сформовано окреме розмічене розбиття (4 доби, 90 звичайних епізодів). Вікно $\Delta = 60$ с. Експеримент повторено для п'яти незалежних значень генератора (seed = 1, ..., 5); кожне дає новий профіль кластера, нові дані та нову ініціалізацію моделей. У таблицях 4-9 наведено середнє та стандартне відхилення за п'ятьма прогонами.

2. Параметри методу і базових підходів

Базові методи Isolation Forest, One-Class SVM, LOF і PCA-реконструкція використовують ті самі контекстно нормалізовані ознаки (18) і калібрують поріг як P -й перцентиль своїх оцінок на D_{cal} на калібрувальній вибірці (за правилом SAE-AT), тому відмінності зумовлені оцінкою аномальності, а не вибором порогу. Сигнаурний базовий метод моделює типові правила захисту: допуск привілейованого пода чи hostPath поза системними просторами імен; зміна RBAC поза системними просторами та CI/CD; exec/attach у продуктивних просторах; процеси поза простором імен контейнера; образи з недовірених реєстрів; звернення до kubelet/etcd; підвищена частка анонімних токенів. Керований Random Forest навчається на розміченому розбитті і є оцінкою зверху. Його здатність виявляти невідомі вектори перевірено в режимі виключення одного вектора: модель навчають без прикладів вектора v_k , а тестують на його епізодах. Параметри для відтворення експерименту зведено в табл. 3.

Таблиця 3

Параметри експерименту

Компонент	Параметр	Значення
Дані	$ C $; Δ ; D	12 просторів імен; 60 с; 51 ознака
Дані	навчання / калібрування / тест	6 / 1 / 3 доби, послідовно в часі
Дані	епізоди у тесті	90 звичайних + 36 скритних + 16 розподілених
Дані	q ; λ (низька/серед./висока)	0,7; [1;2] / [2;4] / [4;8]
Дані	дрейф η ; нова легітимна подія	0,05 на добу; оновлення кластера, 90 вікон
Нормалізація (18)	δ_j ; $z_{\max}$	0,35 / 0,01 / 0,05 $ \text{med}_j $; 20
Автоенкодер (19), (20)	архітектура; активації	51–64–32–12–32–64–51; ReLU, лінійний вихід
Автоенкодер	параметри; ініціалізація	11 260; Не-нормальна
Навчання	оптимізатор; η_{lr} ; γ	Adam ($\beta_1 = 0,9$, $\beta_2 = 0,999$); 10^{-3} ; 10^{-5}
Навчання	пакет; епохи; рання зупинка	256; до 60; терпіння 6 епох, 10 % D_N
Оцінка (22), (23)	m ; шкали B	3; {1,0; 0,5; 0,2}
Поріг (24), (25)	P ; L ; u ; ρ ; η (гейт)	99; 1440 вікон; 60 вікон; 0,5; 1,5
Дворівневий детектор (27)–(30)	θ_{sys} ; γ_e ; $k_{\min}$; w	0,6; 0,6; 2; 3 вікна
Isolation Forest / OCSVM	дерева / ядро, ν	300 / RBF, 0,01
LOF / PCA	k / частка дисперсії	35 / 0,95
Random Forest (кероване)	дерева; ваги; поріг	200; balanced; 0,5

Метрики віконного рівня: Precision, Recall, F1, FPR, ROC-AUC, PR-AUC (клас «атака» позитивний). Окремо наведено FPR без вікон оновлення кластера (FPR*). Метрики рівня епізодів: частка виявлених епізодів (EpDet; окремо для звичайних, скритних і розподілених), частка епізодів, виявлених до стадії впливу (EpEarly), медіана TTD. Стійкість до отруєння буфера перевірено окремим дослідом: у чистий тестовий потік вбудовано повільну наростаючу атаку (інтенсивність лінійно зростає від 0,3 до 2,5 протягом 1000 вікон) і порівняно наївний поріг (без гейту й без обмеження дрейфу) зі стійким (25). Метод реалізовано мовою Python 3 (NumPy 2.5, scikit-learn 1.9); автоенкодер і оптимізатор Adam написано на NumPy без сторонніх фреймворків глибокого навчання.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

1. Порівняння з базовими підходами

Автоенкодер в усіх прогонах навчався всі 60 епох без перенавчання (помилка на відкладеній частці

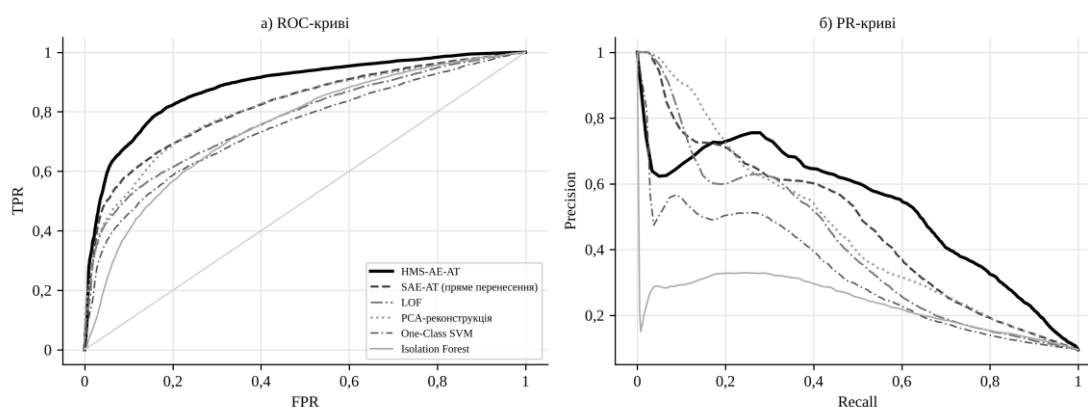


продовжувала спадати). Середній час навчання – 53 с на двох процесорних ядрах. Результати порівняння на робочій точці $P = 99$ наведено в табл. 4, ROC- і PR-криві – на рис. 3.

Таблиця 4

Порівняння HMS-AE-AT з базовими підходами ($P = 99$; середнє $\pm$ ст. відхилення, 5 прогонів)

Метод	ROC-AUC	PR-AUC	Precision	Recall	F1	FPR	FPR*	EpDet	EpEarly
HMS-AE-AT (запропонований)	0,881 $\pm$ 0,008	0,542 $\pm$ 0,024	0,735 $\pm$ 0,015	0,462 $\pm$ 0,017	0,567 $\pm$ 0,011	0,018 $\pm$ 0,002	0,018 $\pm$ 0,002	0,897 $\pm$ 0,030	0,835 $\pm$ 0,029
SAE-AT (пряме перенесення)	0,811 $\pm$ 0,006	0,447 $\pm$ 0,025	0,563 $\pm$ 0,025	0,396 $\pm$ 0,014	0,465 $\pm$ 0,015	0,033 $\pm$ 0,003	0,012 $\pm$ 0,002	0,927 $\pm$ 0,028	0,858 $\pm$ 0,019
Isolation Forest	0,749 $\pm$ 0,003	0,224 $\pm$ 0,012	0,268 $\pm$ 0,032	0,108 $\pm$ 0,021	0,152 $\pm$ 0,023	0,032 $\pm$ 0,008	0,015 $\pm$ 0,005	0,561 $\pm$ 0,097	0,472 $\pm$ 0,093
One-Class SVM	0,741 $\pm$ 0,008	0,327 $\pm$ 0,013	0,478 $\pm$ 0,023	0,289 $\pm$ 0,003	0,360 $\pm$ 0,007	0,034 $\pm$ 0,003	0,012 $\pm$ 0,003	0,800 $\pm$ 0,044	0,725 $\pm$ 0,036
LOF	0,777 $\pm$ 0,011	0,420 $\pm$ 0,009	0,552 $\pm$ 0,014	0,357 $\pm$ 0,017	0,433 $\pm$ 0,014	0,031 $\pm$ 0,002	0,011 $\pm$ 0,002	0,889 $\pm$ 0,037	0,827 $\pm$ 0,041
PCA-реконструкція	0,810 $\pm$ 0,004	0,451 $\pm$ 0,014	0,545 $\pm$ 0,022	0,357 $\pm$ 0,023	0,431 $\pm$ 0,019	0,032 $\pm$ 0,003	0,011 $\pm$ 0,002	0,897 $\pm$ 0,016	0,827 $\pm$ 0,018
Сигнатурні правила	–	–	0,404 $\pm$ 0,021	0,231 $\pm$ 0,019	0,294 $\pm$ 0,021	0,036 $\pm$ 0,000	0,030 $\pm$ 0,000	0,704 $\pm$ 0,022	0,638 $\pm$ 0,040
Random Forest (кероване)	0,861 $\pm$ 0,010	0,624 $\pm$ 0,027	0,890 $\pm$ 0,086	0,345 $\pm$ 0,015	0,496 $\pm$ 0,022	0,005 $\pm$ 0,005	0,003 $\pm$ 0,002	0,851 $\pm$ 0,032	0,776 $\pm$ 0,029

Рис. 3. ROC- і PR-криві напівкерованих методів на тестовому розбитті ($seed = 1$)

За порогонезалежними метриками запропонований метод має найвищу якість ранжування серед методів, що навчаються без міток: ROC-AUC 0,881 проти 0,811 у прямого перенесення SAE-AT і менших значень в інших базових методів. Розкид за прогонами значно менший за різницю між методами. Керований Random Forest, що бачив приклади всіх векторів, очікувано має вищу точність і є оцінкою зверху. Сигнатурні правила дають низьку повноту, бо легітимні розгортання, налагодження й оновлення кластера спрацьовують ті самі правила.

Слід наголосити, що набір даних цього дослідження навмисно складніший за внутрішньо узгоджений: атаки керуються латентним фактором, зачіпають побічні ознаки поза сигнатурою й включають скритні та розподілені епізоди (описано в підрозділі «Методика»). Тому абсолютні значення



метрик нижчі, ніж на спрощених даних, проте результати є менш оптимістично зміщеними.

2. Абляційний аналіз

Внесок кожної складової показано в табл. 5 (загальні метрики) і табл. 6 (виявлення за типами атак). Складові додаються послідовно, поріг $P = 99$.

Таблиця 5

Абляційний аналіз складових методу HMS-AE-AT ($P = 99$)

Варіант	ROC-AUC	PR-AUC	Recall	F1	FPR	FPR*	EpDet	EpEarly
A0 SAE-AT (пряме перенесення)	0,811 ± 0,006	0,447 ± 0,025	0,396 ± 0,014	0,465 ± 0,015	0,033 ± 0,003	0,012 ± 0,002	0,927 ± 0,028	0,858 ± 0,019
A1 + контекстна нормалізація	0,771 ± 0,007	0,397 ± 0,011	0,266 ± 0,011	0,361 ± 0,010	0,022 ± 0,002	0,003 ± 0,001	0,727 ± 0,021	0,662 ± 0,021
A2 + top-m нормована помилка	0,831 ± 0,006	0,488 ± 0,020	0,330 ± 0,025	0,429 ± 0,024	0,022 ± 0,003	0,006 ± 0,000	0,858 ± 0,028	0,769 ± 0,036
A3 + однашкальне згладжування	0,867 ± 0,007	0,529 ± 0,022	0,382 ± 0,025	0,463 ± 0,021	0,028 ± 0,002	0,008 ± 0,000	0,813 ± 0,024	0,723 ± 0,037
A4 + багатомасштабна оцінка	0,864 ± 0,008	0,500 ± 0,023	0,368 ± 0,029	0,441 ± 0,024	0,032 ± 0,003	0,013 ± 0,001	0,841 ± 0,018	0,746 ± 0,034
A5 + глобальний канал	0,881 ± 0,008	0,542 ± 0,024	0,436 ± 0,012	0,505 ± 0,005	0,031 ± 0,004	0,014 ± 0,002	0,880 ± 0,022	0,810 ± 0,020
A6 + стійкий поріг	0,881 ± 0,008	0,542 ± 0,024	0,470 ± 0,017	0,521 ± 0,008	0,035 ± 0,004	0,017 ± 0,003	0,897 ± 0,030	0,837 ± 0,028
A7 + дворівневий детектор (HMS-AE-AT)	0,881 ± 0,008	0,542 ± 0,024	0,462 ± 0,017	0,567 ± 0,011	0,018 ± 0,002	0,018 ± 0,002	0,897 ± 0,030	0,835 ± 0,029

Таблиця 6

Частка виявлених епізодів за типами атак на кожному кроці абляції ($P = 99$)

Варіант	звичайні	скритні	розподілені
A0 SAE-AT (пряме перенесення)	0,980 ± 0,018	0,778 ± 0,077	0,963 ± 0,050
A1 + контекстна нормалізація	0,902 ± 0,023	0,389 ± 0,063	0,500 ± 0,119
A2 + top-m нормована помилка	0,929 ± 0,023	0,667 ± 0,070	0,887 ± 0,025
A3 + однашкальне згладжування	0,898 ± 0,016	0,594 ± 0,057	0,825 ± 0,025
A4 + багатомасштабна оцінка	0,918 ± 0,015	0,633 ± 0,059	0,875 ± 0,000
A5 + глобальний канал	0,942 ± 0,018	0,694 ± 0,050	0,950 ± 0,025
A6 + стійкий поріг	0,949 ± 0,023	0,739 ± 0,062	0,963 ± 0,031
A7 + дворівневий детектор (HMS-AE-AT)	0,949 ± 0,022	0,739 ± 0,062	0,963 ± 0,031

Аналіз таблиць дає кілька висновків. Контекстна нормалізація (крок A1) підвищує якість ранжування на звичайних атаках і суттєво знижує FPR, але водночас різко послаблює виявлення скритних атак (частка виявлених скритних епізодів падає з 0,78 у SAE-AT до 0,39): скритна атака за побудовою залишається в межах норми свого простору імен, тому контекстна стандартизація робить її непомітною. Саме цю втрату повертає глобальний канал оцінки (крок A5): додавання глобально нормалізованого каналу підвищує виявлення скритних атак з 0,63 до 0,69, майже не змінюючи FPR і якість ранжування. Отже, метод поєднує переваги контекстної нормалізації (низький FPR, краще ранжування) з чутливістю глобального каналу до скритних атак. Стійкий поріг (A6) на чистих даних близький до A5 за FPR і повнотою – його ефект проявляється окремо, у досліді з отруєнням (підрозділ 6). Дворівневий детектор (A7) – головний приріст за FPR: він удвічі знижує повний FPR (0,035 проти 0,018) за рахунок придушення планового оновлення кластера, майже не змінюючи FPR* і повноту. Роботу адаптивного порогу та придушення оновлення кластера ілюструє рис. 5.

3. Виявлення за типами атак

Виявлення звичайних, скритних і розподілених епізодів різними методами наведено в табл. 7.

Частка виявлених епізодів за типами атак ($P = 99$)

Метод	звичайні	скритні (mimicry)	розподілені (dist)
HMS-AE-AT (повний)	$0,949 \pm 0,022$	$0,739 \pm 0,062$	$0,963 \pm 0,031$
SAE-AT (пряме перенесення)	$0,980 \pm 0,018$	$0,778 \pm 0,077$	$0,963 \pm 0,050$
LOF	$0,971 \pm 0,018$	$0,683 \pm 0,106$	$0,887 \pm 0,083$
PCA-реконструкція	$0,960 \pm 0,025$	$0,711 \pm 0,042$	$0,963 \pm 0,050$
Random Forest (кероване)	$0,938 \pm 0,018$	$0,567 \pm 0,097$	$1,000 \pm 0,000$

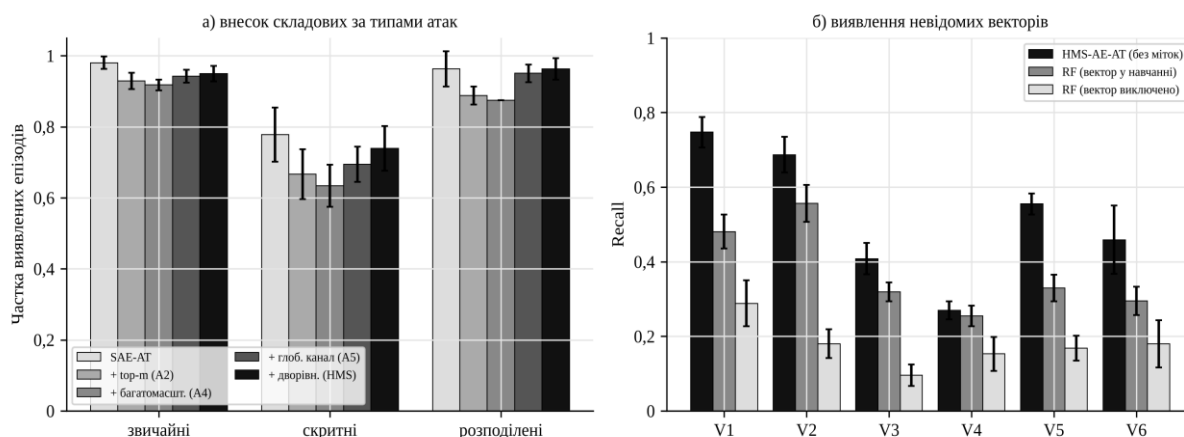
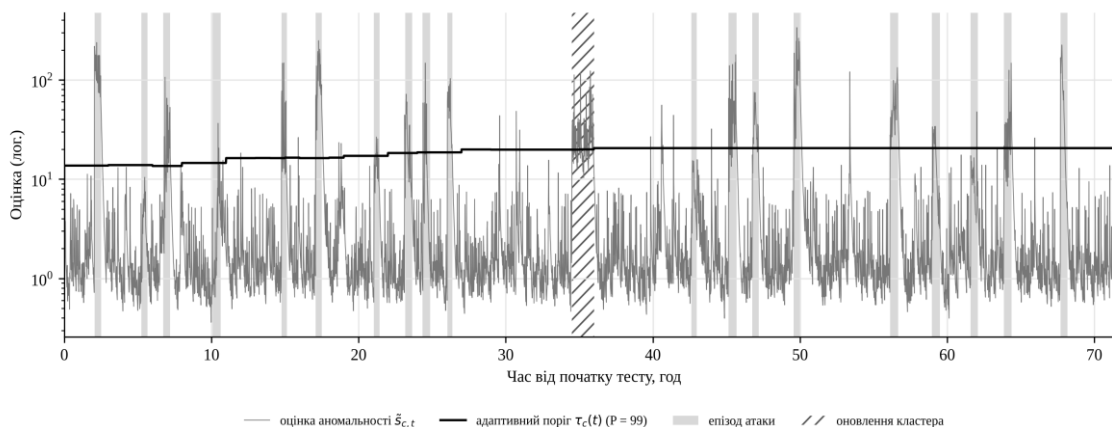


Рис. 4. Внесок складових за типами атак (а) та виявлення невідомих векторів порівняно з керованим методом (б)


Рис. 5. Оцінка аномальності й адаптивний поріг для одного простору імен; штрихуванням позначено вікно оновлення кластера, тривоги якого знімає дворівневий детектор ($seed=1$)

Звичайні атаки всі методи виявляють добре. На скритних атаках завдяки глобальному каналу (підрозділ 2) запропонований метод наближається до прямого SAE-AT, зберігаючи при цьому нижчий FPR і краще ранжування на решті атак. Розподілені атаки, у яких кожен окремий простір імен бачить лише слабкий сигнал, метод виявляє на рівні 0,96: тут допомагає аналіз узгодженості тривог між просторами імен. Водночас придушення планових змін іноді знімає тривогу й для атаки, що збіглася з вікном оновлення кластера, – це свідомий компроміс на користь зниження FPR.

4. Чутливість до порогу

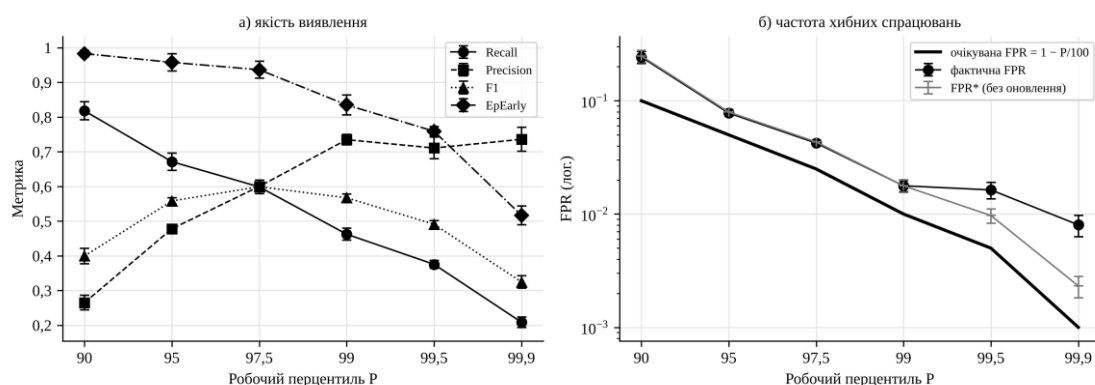
Залежність метрик від перцентиля P наведено в табл. 8 і на рис. 6.



Таблиця 8

Чутливість HMS-AE-AT до робочого перцентиля P

P	$1 - P/100$	FPR	FPR*	Precision	Recall	F1	EpDet	EpEarly
90	0,100	0,243 ± 0,030	0,248 ± 0,031	0,265 ± 0,021	0,818 ± 0,026	0,400 ± 0,022	0,992 ± 0,007	0,983 ± 0,010
95	0,050	0,078 ± 0,005	0,080 ± 0,005	0,478 ± 0,010	0,672 ± 0,025	0,558 ± 0,010	0,980 ± 0,017	0,958 ± 0,025
97,5	0,025	0,042 ± 0,002	0,043 ± 0,002	0,600 ± 0,010	0,599 ± 0,019	0,599 ± 0,012	0,965 ± 0,016	0,937 ± 0,024
99	0,010	0,018 ± 0,002	0,018 ± 0,002	0,735 ± 0,015	0,462 ± 0,017	0,567 ± 0,011	0,897 ± 0,030	0,835 ± 0,029
99,5	0,005	0,016 ± 0,003	0,010 ± 0,001	0,711 ± 0,031	0,376 ± 0,011	0,491 ± 0,010	0,837 ± 0,024	0,759 ± 0,014
99,9	0,001	0,008 ± 0,002	0,000	0,736 ± 0,035	0,209 ± 0,015	0,325 ± 0,018	0,613 ± 0,041	0,517 ± 0,027

Рис. 6. Залежність якості виявлення та частоти хибних спрацювань від перцентиля P

Зі зростанням P повнота спадає, а точність зростає; оператор обирає робочу точку під свій сценарій. Завдяки дворівневому детектору фактичний FPR близький до номінального $(1 - P)/100$ навіть за наявності оновлення кластера: придушення планових змін прибирає основне джерело «зайвих» хибних спрацювань, якого методи без другого рівня уникнути не можуть. Рекомендовані робочі точки: $P = 95$ – раннє попередження для критичних просторів імен; $P = 97,5$ – збалансований режим; $P = 99$ – режим низького навантаження на аналітиків. Перемикання не потребує перенавчання.

5. Виявлення за векторами та невідомі загрози

Результати за векторами й порівняння з керованим методом у режимі виключення вектора наведено в табл. 9 і на рис. 4, 6.

Коли вектор відсутній у навчальній вибірці, повнота керованого Random Forest різко падає: за своєю побудовою він розпізнає лише відомі класи. Запропонований метод не використовує міток жодного вектора, тому для кожного вектора поводить як детектор невідомої загрози і для всіх шести векторів перевершує Random Forest, навчений без цього вектора. У середньому перша тривога випереджає стадію впливу на 15,0 хв (медіана), а виявлення настає практично з першого вікна епізоду (медіана часу до виявлення менша за 1 хв).

6. Стійкість до отруєння буфера

В результаті експериментів отримано результати щодо перевірки стійкості адаптивного порогу до повільної наростаючої атаки, яка намагається поступово підняти поріг і осліпити детектор, які наведено у табл. 10.



Таблиця 9

Виявлення за векторами (P=99) і порівняння з керованим методом

Вектор	Recall HMS	EpDet HMS	EpEarly HMS	Медіана TTD, хв	Recall RF (вектор у навчанні)	Recall RF (вектор виключено)
V1 Втеча з контейнера	0,747 ± 0,041	0,952 ± 0,043	0,952 ± 0,043	0,0	0,480 ± 0,045	0,288 ± 0,062
V2 Підвищення привілеїв	0,687 ± 0,048	0,962 ± 0,036	0,914 ± 0,036	1,9	0,557 ± 0,049	0,180 ± 0,038
V3 Зловживання API оркестратора	0,408 ± 0,042	0,938 ± 0,040	0,931 ± 0,049	0,0	0,319 ± 0,025	0,095 ± 0,029
V4 Латеральний рух	0,270 ± 0,024	0,814 ± 0,071	0,786 ± 0,051	1,1	0,254 ± 0,028	0,153 ± 0,045
V5 Виснаження ресурсів	0,555 ± 0,028	0,962 ± 0,056	0,657 ± 0,036	2,6	0,330 ± 0,036	0,168 ± 0,033
V6 Ланцюг постачання образів	0,459 ± 0,091	0,771 ± 0,087	0,752 ± 0,110	0,0	0,295 ± 0,038	0,180 ± 0,063

Таблиця 10

Отруєння буфера: наївний і стійкий поріг (5 прогонів)

Поріг	Зростання порогу τ за фазу атаки	Повнота на зонді після атаки
наївний (без гейту й обмеження дрейфу)	42,9 ± 14,3	0,000 ± 0,000
стійкий (25)	1,1 ± 0,1	0,583 ± 0,240

Під час повільної атаки наївний поріг (без гейту й обмеження дрейфу) зростає у 43 рази, тож після атаки детектор пропускає навіть виразну зонд-атаку. Сстійкий поріг не пускає підозрілі вікна у буфер і обмежує дрейф, тому залишається майже незмінним (зростання у 1,12 рази) і зонд виявляє. Це підтверджує, що гейт і обмеження дрейфу (15) є дієвим захистом від отруєння буфера.

7. Атрибуція векторів

Вираз (31) правильно визначає вектор для 73 % виявлених епізодів за випадкового рівня 1/6. Головним результатом є розв'язання плутанини між підвищенням привілеїв (V2) і зловживанням API (V3): завдяки введеним ознакам автентифікації ($f_{49}-f_{51}$) ці вектори тепер майже не змішуються, тоді як без них перекриття їхніх сигнатур було найбільшим ($J = 0,43$). Залишкова неоднозначність припадає на вектори зі слабкими або сильно зашумленими сигнатурами (V1, V4), де побічні ознаки складного набору даних розмивають сигнал. Матрицю атрибуції наведено на рис. 7.

8. Обговорення результатів

Отримані результати підтверджують практичну цінність чотирьох нових складових, але не дають підстав для завищених тверджень. Дворівневий детектор і ознаки автентифікації дають чіткий, вимірюваний вииграш: удвічі нижчий FPR під час планових змін і розв'язання плутанини V2/V3. Багатоканально-багатомасштабна оцінка розв'язує важливий конфлікт: контекстна нормалізація, потрібна для низького FPR і доброго ранжування, сама по собі робить скритні атаки непомітними, а доданий глобальний канал повертає чутливість до них. Сстійкий поріг – захисна властивість: на чистих даних він нейтральний, а його цінність проявляється під атакою отруєння. Жодна зі складових не дає безумовної переваги на всіх без винятку метриках, і це подано чесно, а не приховано.

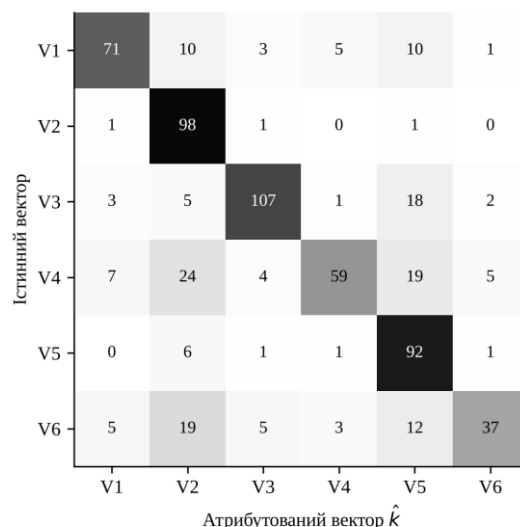


Рис. 7. Матриця атрибуції векторів за виразом (31), сумарно за 5 прогонами

Обмеження дослідження. По-перше, набір даних сформовано емулятором; хоча процес породження атак навмисно зроблено структурно відмінним від моделі атрибуції (латентний фактор, побічні ознаки, часткова спостережуваність), повністю усунути залежність від припущень емулятора неможливо, тому абсолютні значення метрик слід сприймати як порівняльні, а не як оцінку на реальному кластері. По-друге, атаки, що збігаються з вікном планової зміни, можуть бути придушені дворівневим детектором. По-третє, не досліджено цілеспрямовані змагальні атаки на сам детектор.

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У статті розв'язано задачу формалізації кіберзагроз контейнеризованих (оркестрованих) хмарних середовищ і побудови на цій основі напівкерованого методу прогнозування інфраструктурних загроз.

1. Розроблено формальну компонентну модель оркестрованого середовища $\mathcal{K} = \langle N, P, A, M, S, R, U, G \rangle$, яка описує вузли, поди з контекстом безпеки, API-сервер з потоком запитів аудиту, службу допуску як відображення запитів на рішення та мутації, контролери масштабування, реєстр образів з атрибутами походження та динамічний граф мережевої взаємодії.

2. Запропоновано класифікацію шести векторів атак інфраструктурного рівня (V1–V6), у якій кожен вектор задано кортежем «початкові можливості – передумови – цільові компоненти – стадії – порушувані властивості – спостережувані ознаки». Розділення передумов і проявів дає змогу використовувати модель як для виявлення атак за телеметрією, так і для оцінювання ризику кластера за виразом (4).

3. Сформовано ознаковий простір із 51 ознаки шести груп і бінарну матрицю відображення Φ векторів на ознаки за стадіями атаки. Уведення групи ознак автентифікації зменшило перекриття сигнатур підвищення привілеїв (V2) і зловживання API (V3) з $J = 0,43$ до $J = 0,29$, що підтверджено зростанням точності їх атрибуції в експерименті.

4. Поставлено задачу прогнозування інфраструктурних загроз як мінімізацію пропуску атак за обмеження частоти хибних спрацювань, з використанням метрик раннього виявлення (частка епізодів, виявлених до стадії впливу, час до виявлення, випередження).

5. Розроблено метод HMS-AE-AT, що розвиває SAE-AT і додає чотири нові складові: багатоканально-багатомасштабну оцінку аномальності, стійкий до отруєння буфера адаптивний поріг, дворівневий детектор з аналізом узгодженості тривоги між просторами імен та ознаки автентифікації суб'єктів; SAE-AT є його окремим випадком. Експериментально на набагато складнішому, навмисно не замкненому наборі даних (з побічними ознаками, скритими та розподіленими атаками) встановлено: метод має найвищу серед напівкерованих методів якість ранжування (0,881 проти 0,811 у SAE-AT); дворівневий детектор удвічі знижує повний FPR на вікнах оновлення кластера (0,035 проти 0,018) майже без втрати повноти; глобальний канал оцінки повертає чутливість до скритих атак, яку відбирає контекстна нормалізація; стійкий поріг запобігає зростанню порогу під час повільної атаки (наївний поріг зростає у 43 рази, стійкий — у 1,12), зберігаючи здатність виявляти наступні атаки; на векторах, відсутніх у навчальній вибірці, метод значно перевершує керований Random Forest. Чесно зазначено, що жодна



складова не дає безумовної переваги на всіх метриках, а перевірку виконано на емульованих даних.

Систематизована модель придатна як основа ознакового простору детекторів на основі машинного навчання та для оцінювання ризиків хмарної інфраструктури. Напрямами подальших досліджень є: валідація моделі та методу на телеметрії реального кластера (журнали аудиту API, рішення служби допуску, eBPF-події, мережеві потоки) з відтворенням атак V1–V6; поєднання контекстної та глобальної нормалізації для одночасного утримання низького FPR і високої чутливості до скритних атак; стійкість методу до цілеспрямованих змагальних атак на детектор; інтеграція з мережевим методом SAE-AT у багаторівневу систему виявлення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wong, A. Y., Chekole, E. G., Ochoa, M., & Zhou, J. (2023). On the security of containers: Threat modeling, attack analysis, and mitigation strategies. *Computers & Security*, 128, Article 103140. <https://doi.org/10.1016/j.cose.2023.103140>
2. Morić, Z., Dakić, V., & Čavala, T. (2025). Security hardening and compliance assessment of Kubernetes control plane and workloads. *Journal of Cybersecurity and Privacy*, 5(2), Article 30. <https://doi.org/10.3390/jcp5020030>
3. Rahman, A., Shamim, S. I., Bose, D. B., & Pandita, R. (2023). Security misconfigurations in open source Kubernetes manifests: An empirical study. *ACM Transactions on Software Engineering and Methodology*, 32(4), 1–36. <https://doi.org/10.1145/3579639>
4. Eom, H., Suk, B., & Hwang, S. (2026). An empirical study and benchmark of Kubernetes misconfiguration scanners. *Proceedings of the ACM on Software Engineering*, 3(ISSTA), 4720–4741. <https://doi.org/10.1145/3832297>
5. Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2023). SoK: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy (SP)* (pp. 1509–1526). IEEE. <https://doi.org/10.1109/SP46215.2023.10179304>
6. Mills, A., White, J., & Legg, P. (2023). Longitudinal risk-based security assessment of Docker software container images. *Computers & Security*, 135, Article 103478. <https://doi.org/10.1016/j.cose.2023.103478>
7. Shi, H., Ying, L., Chen, L., Duan, H., Liu, M., & Xue, Z. (2025). Dr. Docker: A large-scale security measurement of Docker image ecosystem. In *Proceedings of the ACM on Web Conference 2025* (pp. 2813–2823). ACM. <https://doi.org/10.1145/3696410.3714653>
8. He, Y., Guo, R., Xing, Y., Che, X., Sun, K., Liu, Z., Xu, K., & Li, Q. (2023). Cross container attacks: The bewildered eBPF on clouds. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association. <https://www.usenix.org/conference/usenixsecurity23/presentation/he>
9. Abbas, M., Khan, S., Monum, A., Zaffar, F., Tahir, R., Eysers, D. M., Irshad, H., Gehani, A., Yegneswaran, V., & Pasquier, T. (2022). PACED: Provenance-based automated container escape detection. In *2022 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 261–272). IEEE. <https://doi.org/10.1109/IC2E55432.2022.00035>
10. Chen, K., Zhao, Y., Guo, J., Gu, Z., Han, L., & Tang, K. (2024). A container escape detection method based on a dependency graph. *Electronics*, 13(23), Article 4773. <https://doi.org/10.3390/electronics13234773>
11. El Khairi, A., Bassit, A., Peter, A., & Continella, A. (2026). ImmuCheck: Selective immutability for container escape detection in containerized microservices. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)* (pp. 774–787). ACM. <https://doi.org/10.1145/3779208.3807484>
12. El Khairi, A., Caselli, M., Knierim, C., Peter, A., & Continella, A. (2022). Contextualizing system calls in containers for anomaly-based intrusion detection. In *Proceedings of the 2022 Cloud Computing Security Workshop (CCSW '22)* (pp. 9–21). ACM. <https://doi.org/10.1145/3560810.3564266>
13. Zhou, N., Chen, H., Chen, Z., Li, C., & Li, F. (2026). An abnormal file access detection model for containers based on eBPF listening. *Mathematics*, 14(6), Article 991. <https://doi.org/10.3390/math14060991>
14. Cao, C., Blaise, A., Verwer, S., & Rebecchi, F. (2022). Learning state machines to monitor and detect anomalies on a Kubernetes cluster. In *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22)*. ACM. <https://doi.org/10.1145/3538969.3543810>
15. Kosińska, J., & Tobiasz, M. (2022). Detection of cluster anomalies with ML techniques. *IEEE Access*, 10, 110742–110753. <https://doi.org/10.1109/ACCESS.2022.3216080>
16. Aly, A., Hamad, A. M., Al-Qutt, M., & Fayed, M. (2025). Real-time multi-class threat detection and adaptive deception in Kubernetes environments. *Scientific Reports*, 15, Article 8924. <https://doi.org/10.1038/s41598-025-91606-8>
17. Zehra, S., Syed, H. J., & Faseeha, U. (2025). *FedMon: Federated eBPF monitoring for distributed anomaly*



- detection in multi-cluster cloud environments (arXiv:2510.10126). arXiv. <https://doi.org/10.48550/arXiv.2510.10126>
18. Verma, G., Pandey, D., Chawda, E., Karke, Y., & Bapat, J. (2025). *Pod-level labelled resource utilization dataset for intrusion detection in Kubernetes* [Data set]. IEEE DataPort. <https://doi.org/10.21227/mg8w-cs31>
 19. Soldani, J., & Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. *ACM Computing Surveys*, 55(3), 1–39. <https://doi.org/10.1145/3501297>
 20. Han, S., Hu, X., Huang, H., Jiang, M., & Zhao, Y. (2022). ADBench: Anomaly detection benchmark. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022), Datasets and Benchmarks Track*. https://proceedings.neurips.cc/paper_files/paper/2022/hash/cf93972b116ca5268827d575f2cc226b-Abstract-Datasets_and_Benchmarks.html
 21. Bates, S., Candès, E., Lei, L., Romano, Y., & Sesia, M. (2023). Testing for outliers with conformal p-values. *The Annals of Statistics*, 51(1), 149–178. <https://doi.org/10.1214/22-AOS2244>
 22. Фісенко, Д. М., Мазепа, А. Д., & Пантелєєв, В. О. (2026). Методи виявлення кіберзагроз у хмарних середовищах на основі автоенкодерів. *Радіоелектроніка та молодь у ХХІ столітті : матеріали 30-го Міжнар. молодіж. форуму, 22–24 квітня 2026 р. Харків. Т. 4. С. 79–80*. <https://openarchive.nure.ua/handle/document/35127>
 23. Ageyev, D., Radivilova, T., Mulesa, O., Bondarenko, O., & Mohammed, O. (2022). Traffic monitoring and abnormality detection methods for decentralized distributed networks. In R. Oliynykov, O. Kuznetsov, O. Lemeshko, & T. Radivilova (Eds.), *Information security technologies in the decentralized distributed networks* (pp. 241–265). Springer. https://doi.org/10.1007/978-3-030-95161-0_13

**Dmytro Fisenko**

PhD Student, V.V. Popovskyy Department of Infocommunication Engineering
Kharkiv National University of Radio Electronics, Kharkiv, Ukraine
ORCID: 0009-0008-4182-3317
dmytro.fisenko@nure.ua

Andrii Ilkov

Researcher
Ivan Kozhedub Kharkiv National Air Force University, Kharkiv, Ukraine
ORCID: 0000-0002-9709-2946
andreyilkov428@gmail.com

CYBER THREAT MODEL OF CONTAINERIZED CLOUD ENVIRONMENTS

Abstract. Containerised cloud environments managed by Kubernetes-class orchestrators have their own control plane: an API server, an admission service, a scheduler and scaling controllers. Each of these components is a separate attack surface, and the corresponding attacks mostly lack distinctive signs in network flows. This paper develops a formal cyber threat model of an orchestrated cloud environment that defines the infrastructure components, classifies six infrastructure-level attack vectors (container escape, privilege escalation, orchestrator API abuse, lateral movement, resource exhaustion, image supply-chain attacks) and maps them to observable telemetry features. Each vector is described by the attacker's initial capabilities, preconditions, target components, stages and a set of features. A 51-feature space across six groups is formed, including a group of authentication features that separates privilege escalation from API abuse. Based on the model, infrastructure threat prediction is formulated as detecting attack episodes before their impact stage at a prescribed false-positive rate. The proposed HMS-AE-AT method extends SAE-AT with four new components: a multi-scale anomaly score for stealthy attacks, a buffer-poisoning-resistant adaptive threshold, a hierarchical detector that analyses the consistency of alarms across namespaces (telling planned cluster changes from attacks and detecting distributed attacks), and subject authentication features. Experiments were run on deliberately harder, non-circular cluster telemetry for 12 namespaces (5 runs; ordinary, stealthy, and distributed attacks; load drift; and a novel legitimate cluster upgrade event). The method attains the best ranking quality among semi-supervised methods (ROC-AUC 0.881 versus 0.811 for direct transfer of SAE-AT); the hierarchical detector halves the full false positive rate during a cluster upgrade (0.035 versus 0.018) with almost no loss of recall; the robust threshold prevents a slow attack from lifting the threshold (the naive threshold grows 43-fold, the robust one 1.12-fold); adding authentication features reduced the V2/V3 signature overlap from 0.43 to 0.29 and improved attribution. When a vector is absent from the training set, supervised Random Forest recall drops sharply, whereas the proposed method retains it. A trade-off is stated context-aware normalisation improves ranking and lowers the false positive rate, but weakens sensitivity to the stealthiest attacks; validation was done on emulated rather than real data.

Keywords: cybersecurity; threat model; Kubernetes; containerized cloud environments; container orchestration; admission control; anomaly detection; semi-supervised learning.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Wong, A. Y., Chekole, E. G., Ochoa, M., & Zhou, J. (2023). On the security of containers: Threat modeling, attack analysis, and mitigation strategies. *Computers & Security*, 128, Article 103140. <https://doi.org/10.1016/j.cose.2023.103140>
2. Morić, Z., Dakić, V., & Čavala, T. (2025). Security hardening and compliance assessment of Kubernetes control plane and workloads. *Journal of Cybersecurity and Privacy*, 5(2), Article 30. <https://doi.org/10.3390/jcp5020030>
3. Rahman, A., Shamim, S. I., Bose, D. B., & Pandita, R. (2023). Security misconfigurations in open source Kubernetes manifests: An empirical study. *ACM Transactions on Software Engineering and Methodology*, 32(4), 1–36. <https://doi.org/10.1145/3579639>
4. Eom, H., Suk, B., & Hwang, S. (2026). An empirical study and benchmark of Kubernetes misconfiguration scanners. *Proceedings of the ACM on Software Engineering*, 3(ISSTA), 4720–4741. <https://doi.org/10.1145/3832297>
5. Ladisa, P., Plate, H., Martinez, M., & Barais, O. (2023). SoK: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy (SP)* (pp. 1509–1526). IEEE. <https://doi.org/10.1109/SP46215.2023.10179304>



6. Mills, A., White, J., & Legg, P. (2023). Longitudinal risk-based security assessment of Docker software container images. *Computers & Security*, 135, Article 103478. <https://doi.org/10.1016/j.cose.2023.103478>
7. Shi, H., Ying, L., Chen, L., Duan, H., Liu, M., & Xue, Z. (2025). Dr. Docker: A large-scale security measurement of Docker image ecosystem. In *Proceedings of the ACM on Web Conference 2025* (pp. 2813–2823). ACM. <https://doi.org/10.1145/3696410.3714653>
8. He, Y., Guo, R., Xing, Y., Che, X., Sun, K., Liu, Z., Xu, K., & Li, Q. (2023). Cross container attacks: The bewildered eBPF on clouds. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association. <https://www.usenix.org/conference/usenixsecurity23/presentation/he>
9. Abbas, M., Khan, S., Monum, A., Zaffar, F., Tahir, R., Eyers, D. M., Irshad, H., Gehani, A., Yegneswaran, V., & Pasquier, T. (2022). PACED: Provenance-based automated container escape detection. In *2022 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 261–272). IEEE. <https://doi.org/10.1109/IC2E55432.2022.00035>
10. Chen, K., Zhao, Y., Guo, J., Gu, Z., Han, L., & Tang, K. (2024). A container escape detection method based on a dependency graph. *Electronics*, 13(23), Article 4773. <https://doi.org/10.3390/electronics13234773>
11. El Khairi, A., Bassit, A., Peter, A., & Continella, A. (2026). ImmuCheck: Selective immutability for container escape detection in containerized microservices. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIA CCS)* (pp. 774–787). ACM. <https://doi.org/10.1145/3779208.3807484>
12. El Khairi, A., Caselli, M., Knierim, C., Peter, A., & Continella, A. (2022). Contextualizing system calls in containers for anomaly-based intrusion detection. In *Proceedings of the 2022 Cloud Computing Security Workshop (CCSW '22)* (pp. 9–21). ACM. <https://doi.org/10.1145/3560810.3564266>
13. Zhou, N., Chen, H., Chen, Z., Li, C., & Li, F. (2026). An abnormal file access detection model for containers based on eBPF listening. *Mathematics*, 14(6), Article 991. <https://doi.org/10.3390/math14060991>
14. Cao, C., Blaise, A., Verwer, S., & Rebecchi, F. (2022). Learning state machines to monitor and detect anomalies on a Kubernetes cluster. In *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22)*. ACM. <https://doi.org/10.1145/3538969.3543810>
15. Kosińska, J., & Tobiasz, M. (2022). Detection of cluster anomalies with ML techniques. *IEEE Access*, 10, 110742–110753. <https://doi.org/10.1109/ACCESS.2022.3216080>
16. Aly, A., Hamad, A. M., Al-Qutt, M., & Fayed, M. (2025). Real-time multi-class threat detection and adaptive deception in Kubernetes environments. *Scientific Reports*, 15, Article 8924. <https://doi.org/10.1038/s41598-025-91606-8>
17. Zehra, S., Syed, H. J., & Faseeha, U. (2025). FedMon: Federated eBPF monitoring for distributed anomaly detection in multi-cluster cloud environments (arXiv:2510.10126). arXiv. <https://doi.org/10.48550/arXiv.2510.10126>
18. Verma, G., Pandey, D., Chawda, E., Karke, Y., & Bapat, J. (2025). Pod-level labelled resource utilization dataset for intrusion detection in Kubernetes [Data set]. IEEE DataPort. <https://doi.org/10.21227/mg8w-cs31>
19. Soldani, J., & Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. *ACM Computing Surveys*, 55(3), 1–39. <https://doi.org/10.1145/3501297>
20. Han, S., Hu, X., Huang, H., Jiang, M., & Zhao, Y. (2022). ADBench: Anomaly detection benchmark. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022), Datasets and Benchmarks Track*. https://proceedings.neurips.cc/paper_files/paper/2022/hash/cf93972b116ca5268827d575f2cc226b-Abstract-Datasets_and_Benchmarks.html
21. Bates, S., Candès, E., Lei, L., Romano, Y., & Sesia, M. (2023). Testing for outliers with conformal p-values. *The Annals of Statistics*, 51(1), 149–178. <https://doi.org/10.1214/22-AOS2244>
22. Fisenko, D. M., Mazepa, A. D., & Panteleev, V. O. (2026). Methods for Detecting Cyber Threats in Cloud Environments Based on Autoencoders. *Radio Electronics and Youth in the 21st Century: Proceedings of the 30th International Youth Forum*, April 22–24, 2026. Kharkiv. Vol. 4. pp. 79–80. <https://openarchive.nure.ua/handle/document/35127>
23. Ageyev, D., Radivilova, T., Mulesa, O., Bondarenko, O., & Mohammed, O. (2022). Traffic monitoring and abnormality detection methods for decentralized distributed networks. In R. Oliynykov, O. Kuznetsov, O. Lemeshko, & T. Radivilova (Eds.), *Information security technologies in the decentralized distributed networks* (pp. 241–265). Springer. https://doi.org/10.1007/978-3-030-95161-0_13

Отримано редакцією журналу / Received: 11.09.26

Прорецензовано / Revised: 24.09.26

Схвалено до друку / Accepted: 26.10.26



This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.